Constraint Satisfaction Problems

Chapter 6

Pearson

Copyright © 2021 Pearson Education, Inc. All Rights Reserved

Outline

- Introduction
- Defining Constraint Satisfaction Problems (CSP)
- CSP examples
- Backtracking search for CSPs
- Local search for CSPs
- Problem structure and problem decomposition

© 2021 Pearson Education Ltd.

Introduction

Chapters 3 through 5 treat every state as a "black box." The search algorithm knows how to generate successors and how to test whether a state is a goal, but it knows nothing about what is inside the state.

Now, we open the box. We go from an atomic representation of a state to a factored representation.

Why does this matter? Because once you can see inside the state, you can reason about it in ways that were impossible before. If you assign a value to one variable and that assignment violates a constraint, you can immediately rule out large portions of the search space without exploring them.

Think of it as the difference between finding your keys by searching every room in the house one by one, versus knowing they must be in a room with a door you opened today. The problem is the same; the information you use to

Pearson © 2021 Pearson Education Ltd.

Example: Motivation

Suppose you are scheduling three meetings (A, B, and C) into three time slots (9am, 10am, 11am). The conditions are:

Meeting A must happen before meeting B. The manager who attends both B and C cannot be in two places at once, so B and C cannot be at the same time. Meeting C must be at 10 am or later.

So, 3 meetings that may happen at 3 times slots: $3^3 = 27$, but you check the constraints, and realize you can discard many scenarios. A blind search will check for(A at 9 am, B at 9 am, and C at 9 am)

If you assign A to 10 am, you know immediately that B must be at 11 am (because B must come after A, and 11 am is the only later slot). And since B is at 11 am, C cannot also be at 11 am, so C must be at 10 am.

Defining Constraint Satisfaction Problems

A constraint satisfaction problem (CSP) consists of three components, X, D, and C:

- X is a set of variables, $\{X_1 \ldots X_n\}$.
- D is a set of domains, $\{D_1, \ldots, D_n\}$, one for each variable
- C is a set of constraints that specify an allowable combination of values

CSPs deal with assignments of values to variables.

- A complete assignment is one in which every variable is assigned a value, and a solution to a CSP is a consistent, complete assignment.
- A partial assignment is one that leaves some variables unassigned.

© 2021 Pearson Education Ltd.

Constraint satisfaction problems (CSPs)

Standard search problem:
state is a "black box"—any old data structure that supports goal test, eval, successor

CSP:
state is defined by variables X_i with values from the domain D_i

goal test is a set of constraints specifying allowable combinations of values for subsets of variables

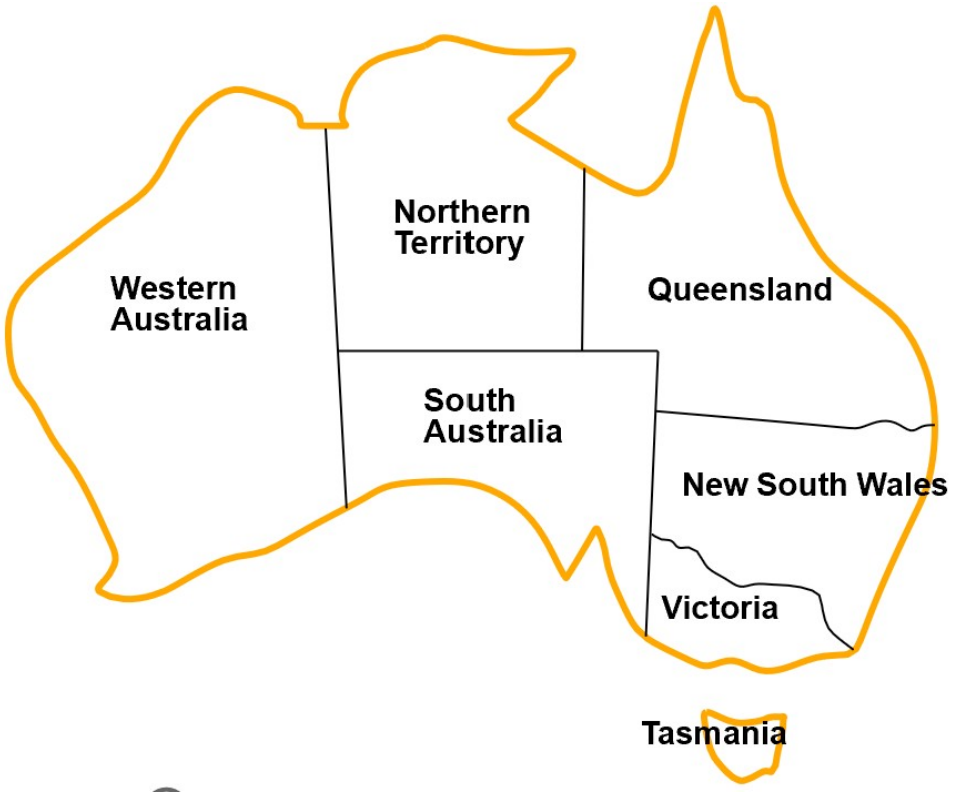
Simple example of a formal representation language

Allows useful general-purpose algorithms with more power than standard search algorithms

© 2021 Pearson Education Ltd.

Example: Map-Coloring

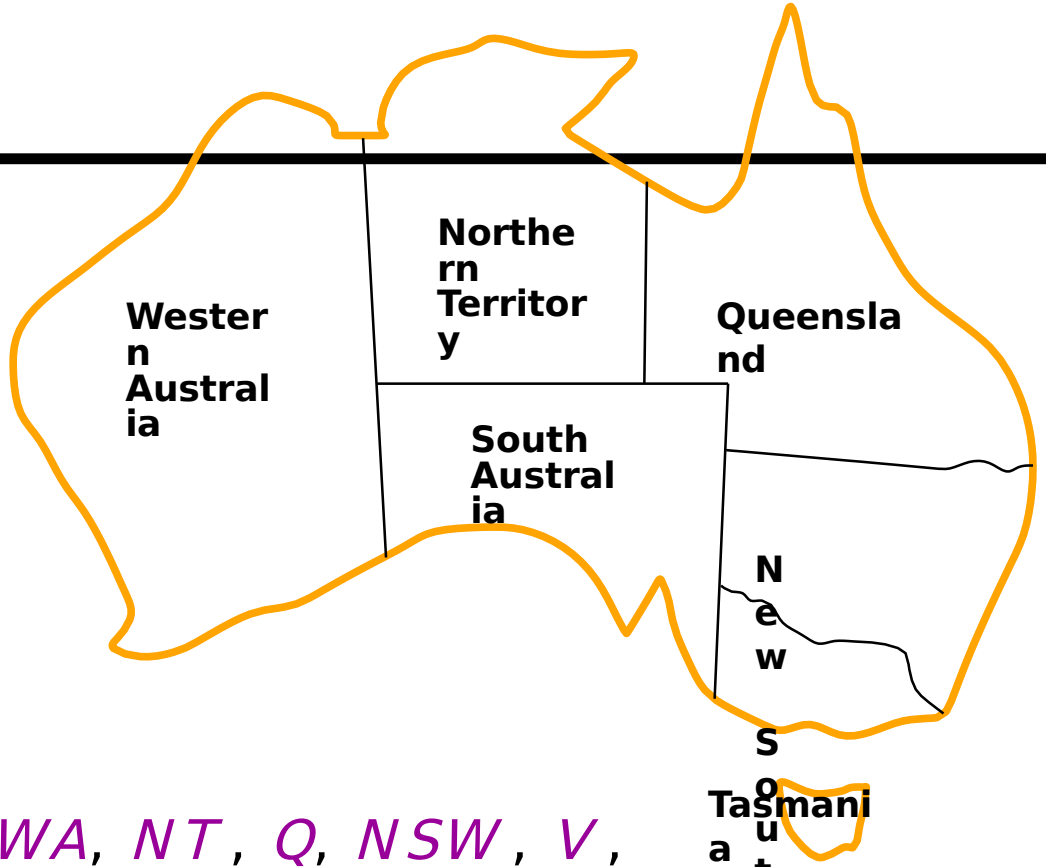
We are looking at a map of Australia showing each of its states and territories. The task consists of coloring each region either red, green, or blue in such a way that no two neighboring regions have the same color. Formulate this as a CSP.



Pearson

© 2021 Pearson Education Ltd.

Example: Map-Coloring



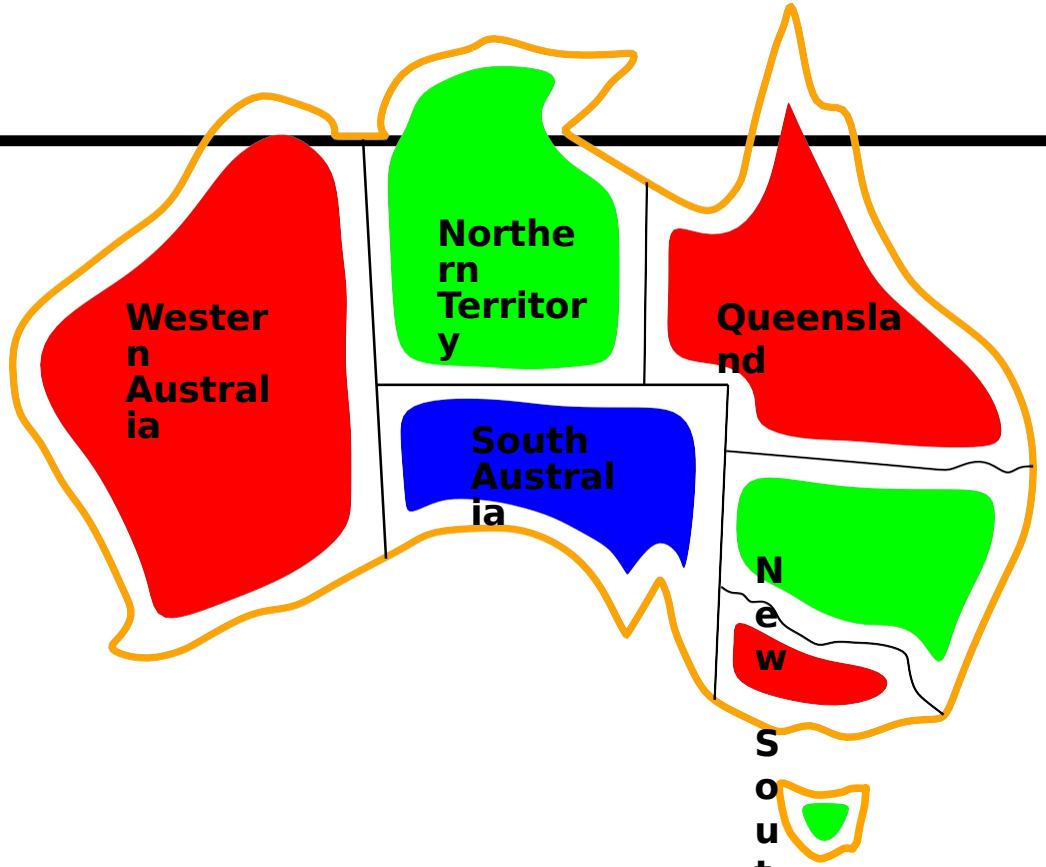
Variables WA, NT, Q, NSW, V, SA, T

Domains: $D_i = \{red, green, blue\}$

Constraints: adjacent regions must have different colors e.g., $WA \neq NT$ (if the language allows this), or

$(WA, NT) \in \{(red, green), (red, blue), (green, red), (green, blue), \ldots\}$

Example: Map-Coloring contd.



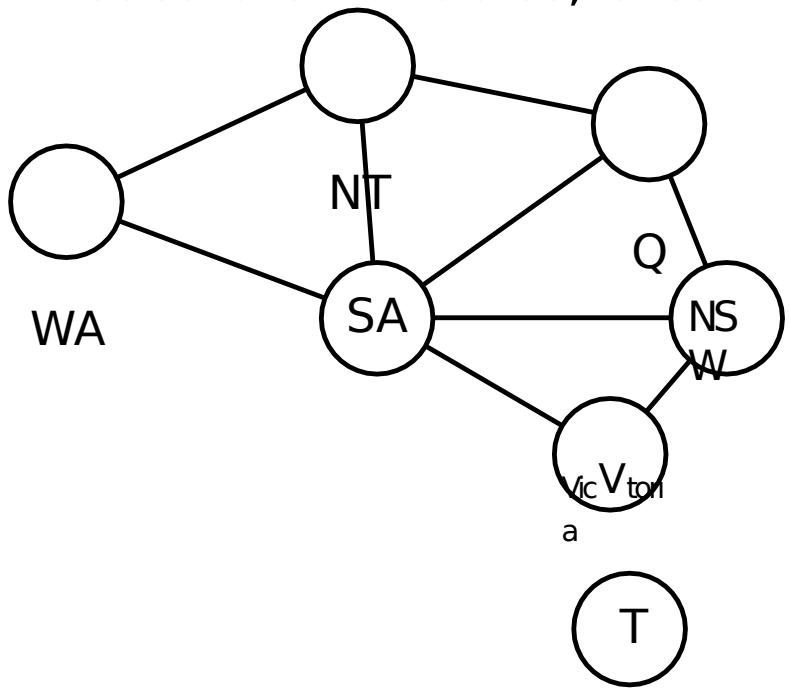
Solutions are assignments satisfying all constraints, e.g.,
{ WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green}

© 2021 Pearson Education Ltd.

Constraint graph

Binary CSP: each constraint relates at most two variables

Constraint graph: nodes are variables, arcs show constraints



General-purpose CSP algorithms use the graph structure
to speed up search. E.g., Tasmania is an independent subproblem!

© 2021 Pearson Education Ltd.

Varieties of CSPs

Discrete variables

finite domains; size d ⇒ $O(d^n)$ complete assignments

- e.g., Boolean CSPs, incl. Boolean satisfiability (NP-complete) infinite domains (integers, strings, etc.)
- e.g., job scheduling, variables are start/end days for each job
- need a constraint language, e.g., $StartJob_1 + 5 \leq StartJob_3$
- linear constraints solvable, nonlinear undecidable

Continuous variables

- e.g., start/end times for Hubble Telescope observations
- linear constraints solvable in poly time by LP methods

© 2021 Pearson Education Ltd.

Varieties of constraints

Unary constraints involve a single variable, e.g., *SA /= green*

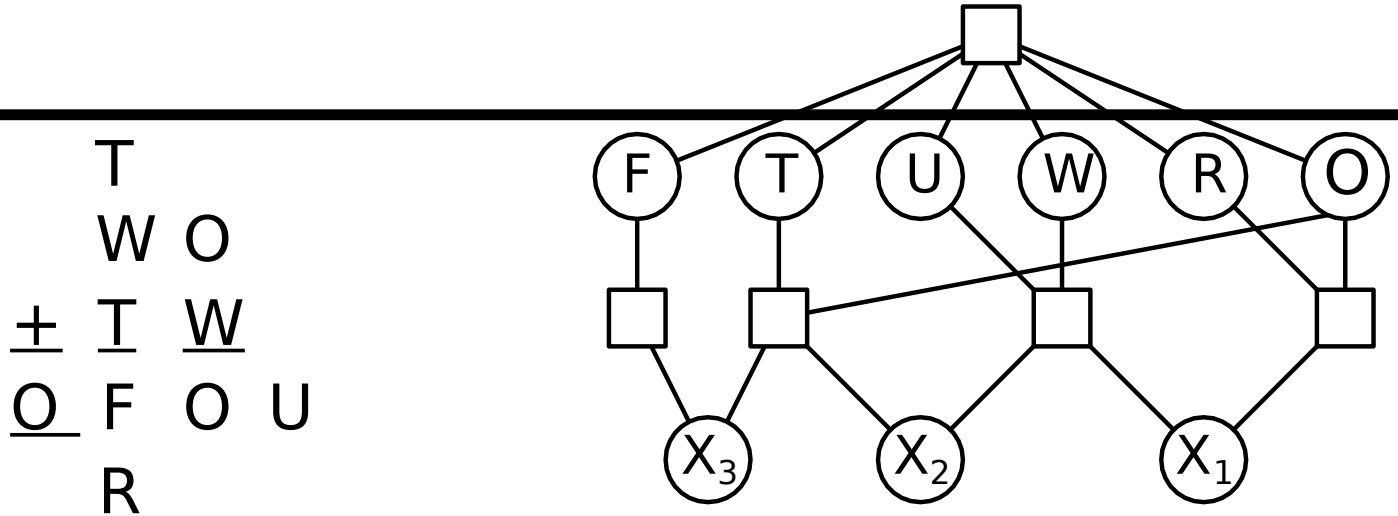
Binary constraints involve pairs of variables, e.g., *SA /= WA*

Higher-order constraints involve 3 or more variables, e.g., cryptarithmetic column constraints

Preferences (soft constraints), e.g., *red* is better than *green*
often representable by a cost for each variable assignment
→ constrained optimization problems

© 2021 Pearson Education Ltd.

Example: Cryptarithmetic



Variables: $F\ T\ U\ W\ R\ O\ X_1\ X_2\ X_3$

Domains: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Constraints

$alldiff(F, T, U, W, R, O)$

$O + O = R + 10 \cdot X_1$, etc.

© 2021 Pearson Education Ltd.

Real-world CSPs

Assignment problems
e.g., who teaches what class

Timetabling problems
e.g., which class is offered when and where?

Hardware configuration

Spreadsheets

Transportation

scheduling Factory

scheduling

Floorplanning

Notice that many real-world problems involve real-valued variables

© 2021 Pearson Education Ltd.

Standard search formulation (incremental)

Let's start with the straightforward, dumb approach, then fix it

States are defined by the values assigned so far

- Initial state: the empty assignment, { }
- Successor function: assign a value to an unassigned variable that does not conflict with current assignment.
  ⇒ fail if no legal assignments (not fixable!)
- Goal test: the current assignment is complete

1) This is the same for all CSPs!
2) Every solution appears at depth n with n variables
   ⇒ use depth-first search
3) Path is irrelevant, so can also use complete-state formulation

© 2021 Pearson Education Ltd.

Backtracking search

Variable assignments are commutative, i.e.,

[$WA = red$ then $NT = green$] same as [$NT = green$ then $WA = red$]

Only need to consider assignments to a single variable at each node

$\Rightarrow$ $b = d$ and there are d^n leaves

Depth-first search for CSPs with single-variable assignments is called backtracking search

Backtracking search is the basic uninformed

algorithm for CSPs Can solve n-queens for $n \approx 25$

© 2021 Pearson Education Ltd.

Backtracking search

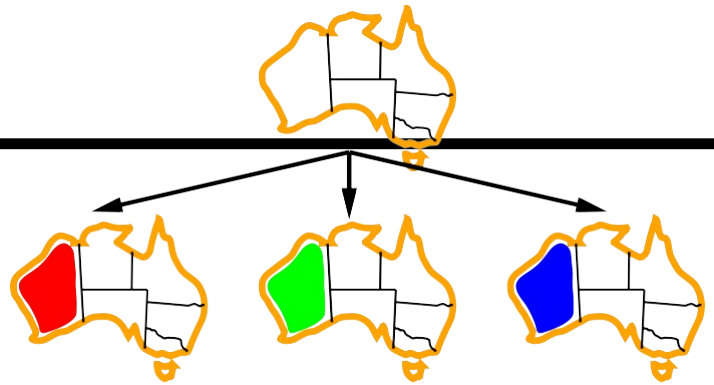
```
function Backtracking-Search(csp) returns solution/failure
  return Recursive-Backtracking({ }, csp)

function Recursive-Backtracking(assignment, csp) returns
soln/failure
  if assignment is complete then return assignment
  var ← Select-Unassigned-Variable(Variables[csp],
  assignment, csp)
  for each value in Order-Domain-Values(var, assignment,
  csp) do
      if value is consistent with assignment given
      Constraints[csp] then
          add {var = value} to assignment
          result ← Recursive-Backtracking(assignment, csp)
          if result /= failure then return result
          remove {var = value} from assignment
  return failure
```


© 2021 Pearson Education Ltd.

Backtracking example

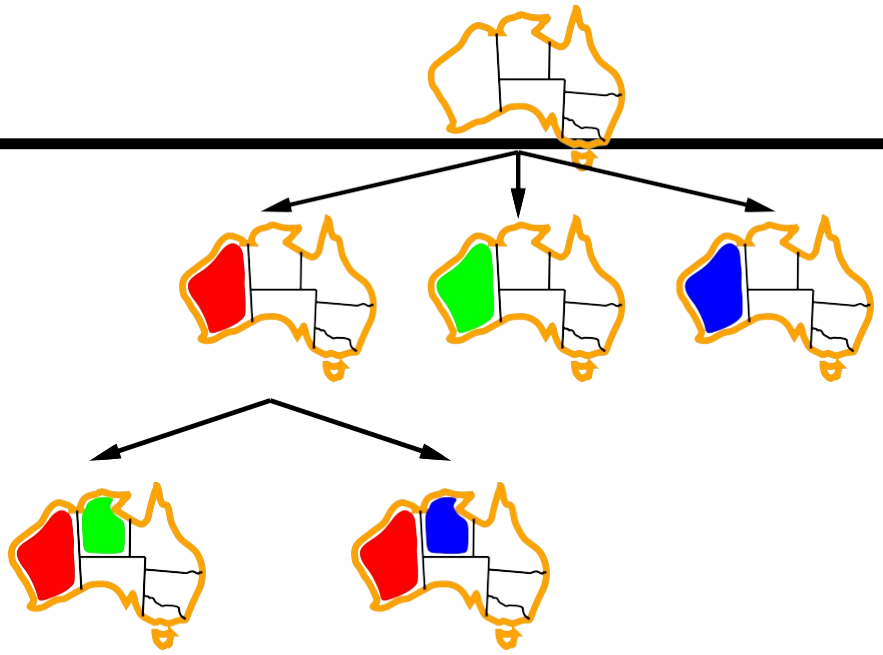
© 2021 Pearson Education Ltd.

Backtracking example



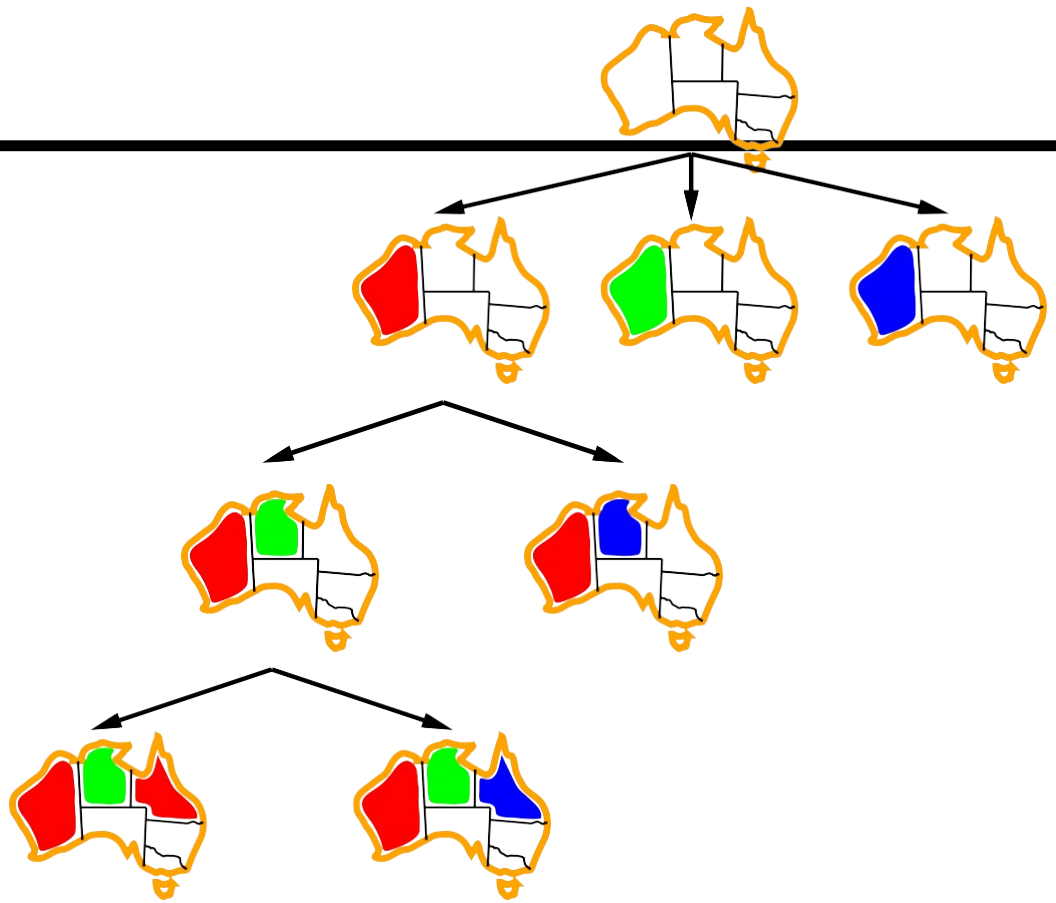
© 2021 Pearson Education Ltd.

Backtracking example



© 2021 Pearson Education Ltd.

Backtracking example



© 2021 Pearson Education Ltd.

Improving backtracking efficiency

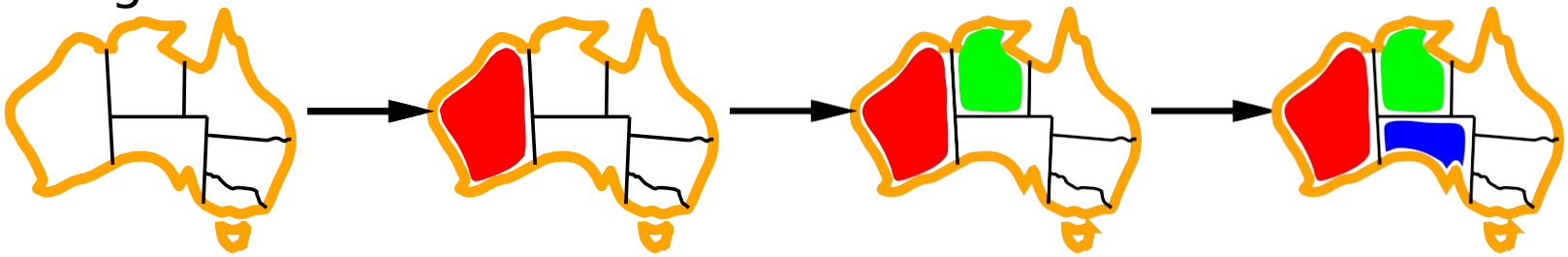
General-purpose methods can give huge gains in speed:

1. Which variable should be assigned next?
2. In what order should its values be tried?
3. Can we detect inevitable failure early?
4. Can we take advantage of problem structure?

© 2021 Pearson Education Ltd.

Minimum remaining values

Minimum remaining values (MRV):
choose the variable with the fewest legal values



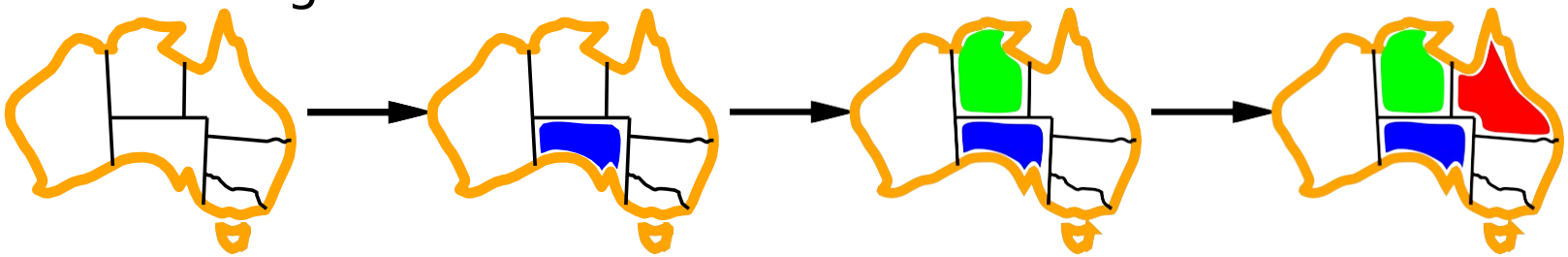
© 2021 Pearson Education Ltd.

Degree heuristic

Tie-breaker among MRV variables

Degree heuristic:

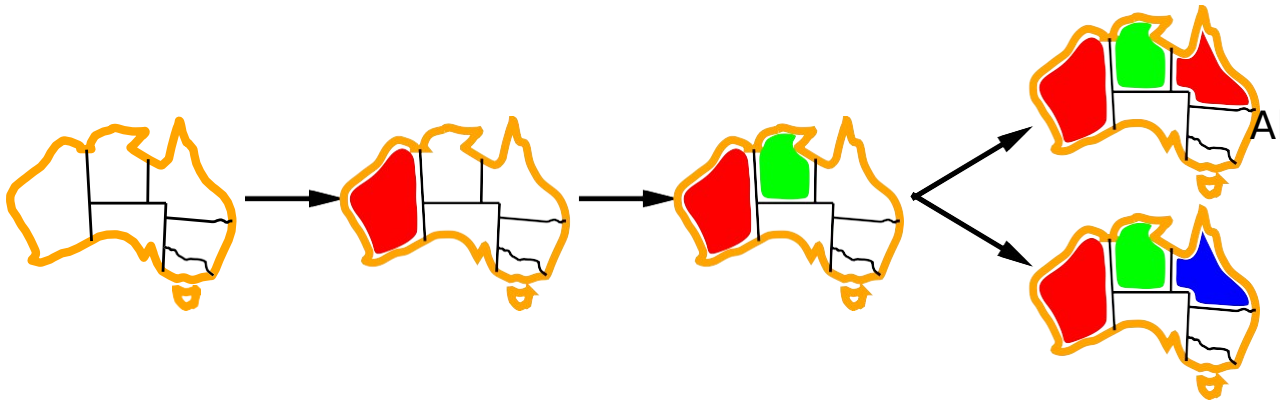
choose the variable with the most constraints on remaining variables



© 2021 Pearson Education Ltd.

Least constraining value

Given a variable, choose the least constraining value:
the one that rules out the fewest values in the remaining variables

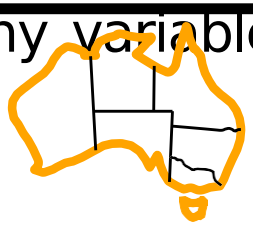


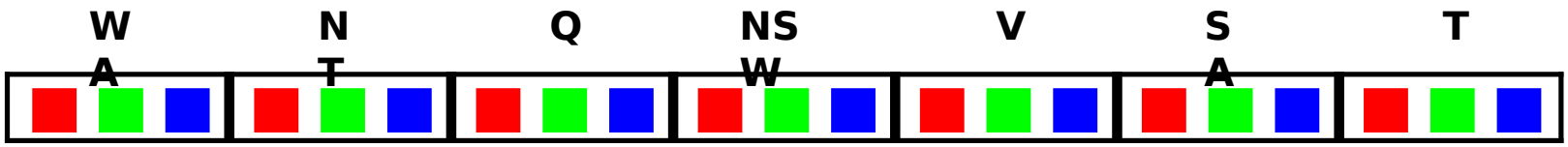
Combining these heuristics makes 1000 queens feasible

© 2021 Pearson Education Ltd.

Forward checking

Idea: Keep track of remaining legal values for unassigned variables Terminate search when any variable has no legal values

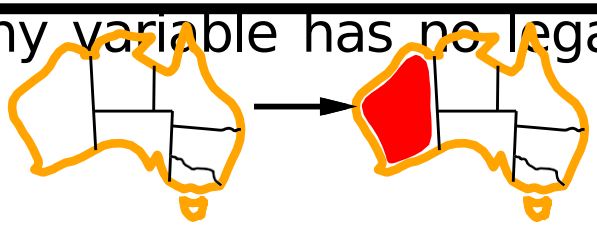


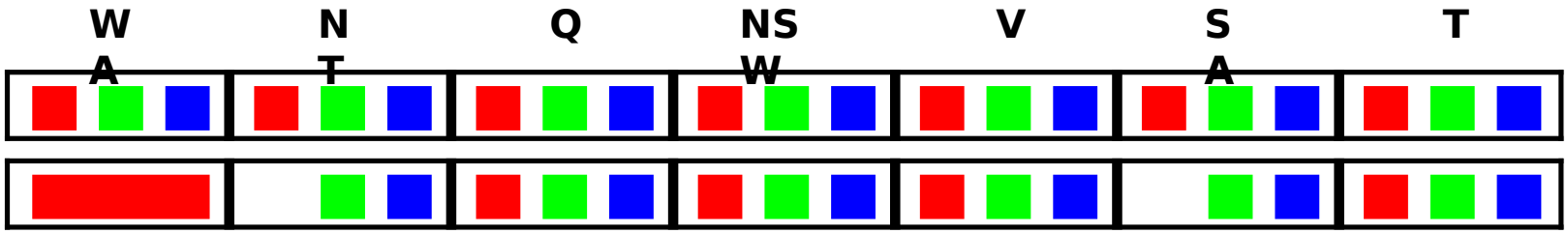


© 2021 Pearson Education Ltd.

Forward checking

Idea: Keep track of remaining legal values for unassigned variables Terminate search when any variable has no legal values

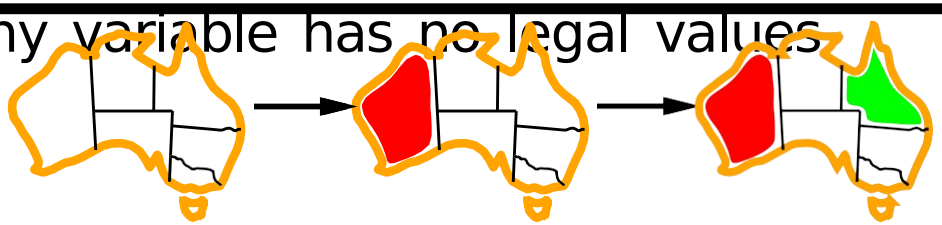


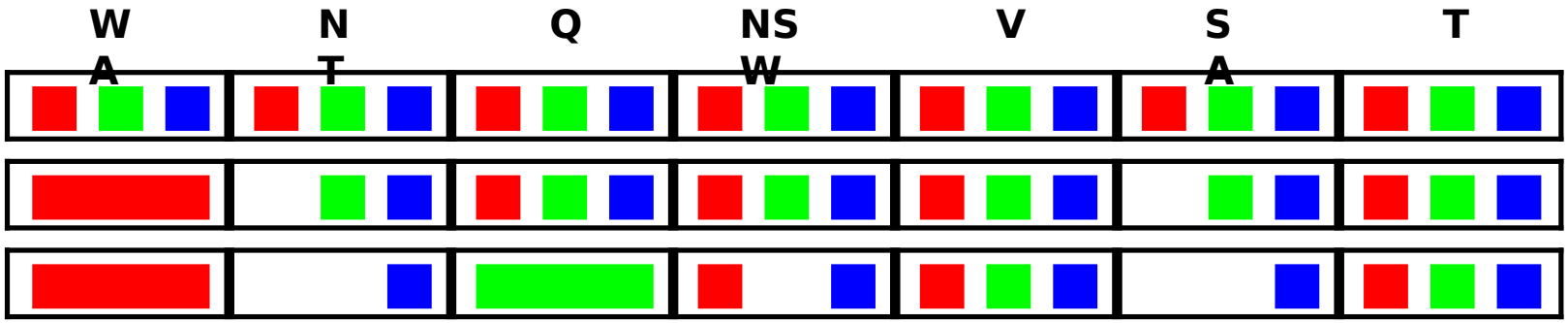


© 2021 Pearson Education Ltd.

Forward checking

Idea: Keep track of remaining legal values for unassigned variables Terminate search when any variable has no legal values

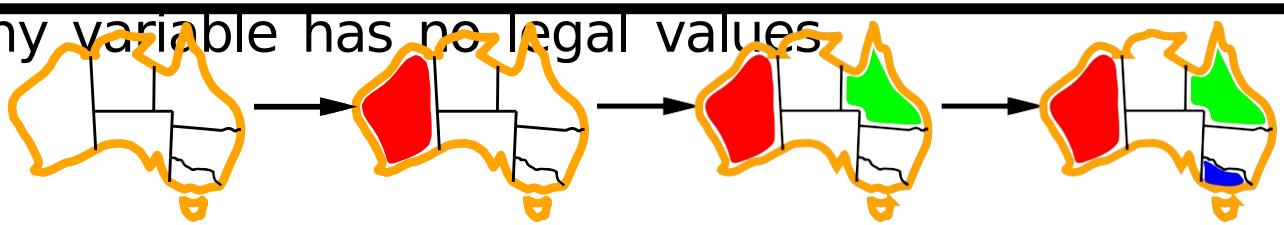


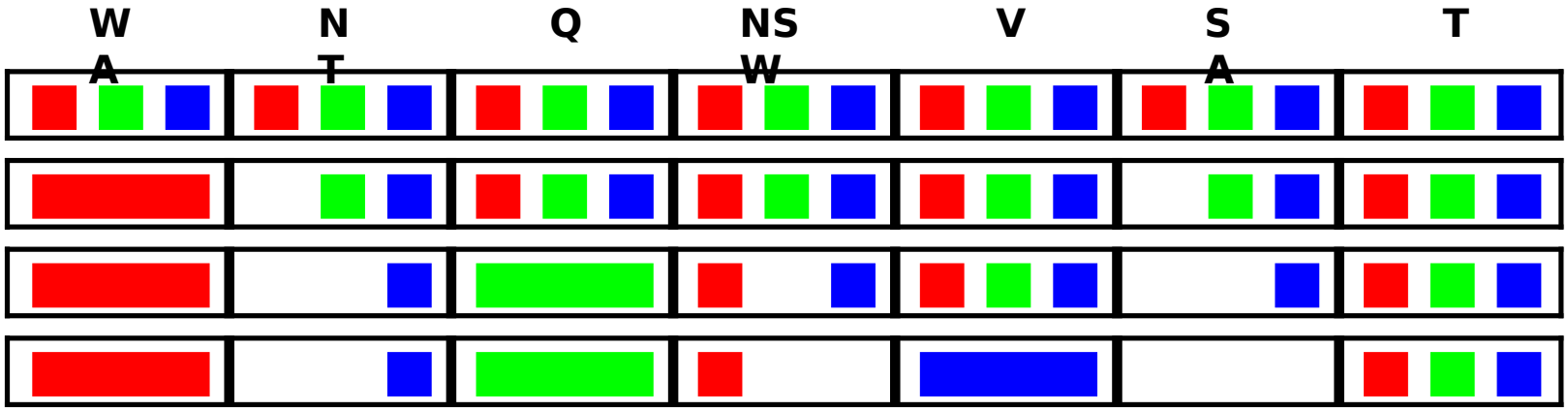


© 2021 Pearson Education Ltd.

Forward checking

Idea: Keep track of remaining legal values for unassigned variables Terminate search when any variable has no legal values

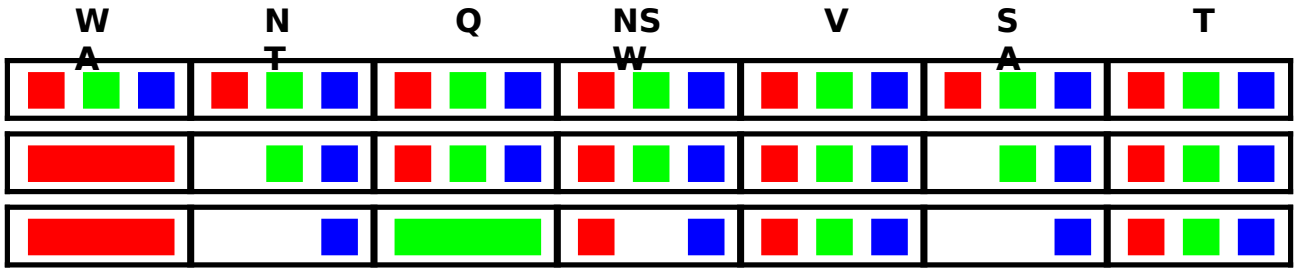




© 2021 Pearson Education Ltd.

Constraint propagation

Forward checking propagates information from assigned to unassigned vari- ables, but doesn't provide early detection for all failures:



NT and *SA* cannot both be blue!

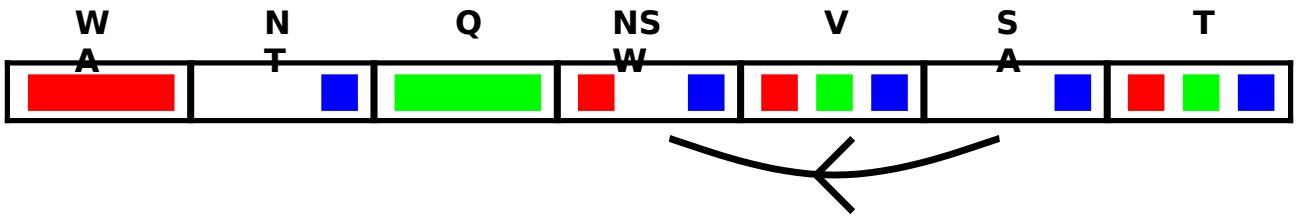
Constraint propagation repeatedly enforces constraints locally

© 2021 Pearson Education Ltd.

Arc consistency

Simplest form of propagation makes each arc consistent

$X \rightarrow Y$ is consistent iff
for every value x of X there is some allowed y

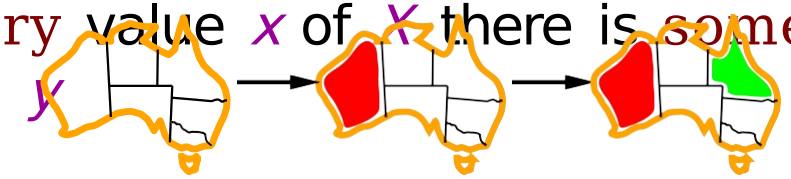


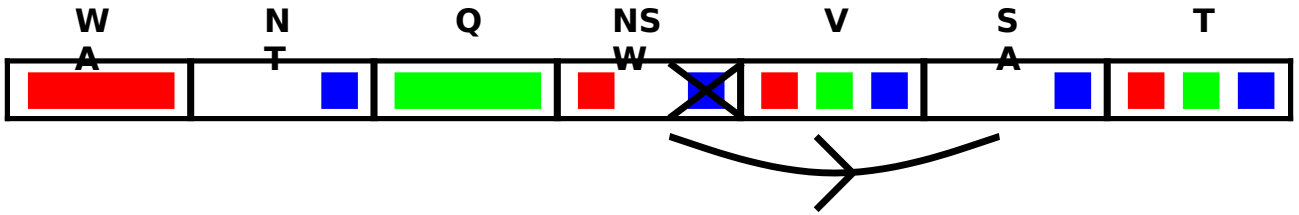
© 2021 Pearson Education Ltd.

Arc consistency

Simplest form of propagation makes each arc consistent

$X \rightarrow Y$ is consistent iff
for every value x of X there is some allowed y



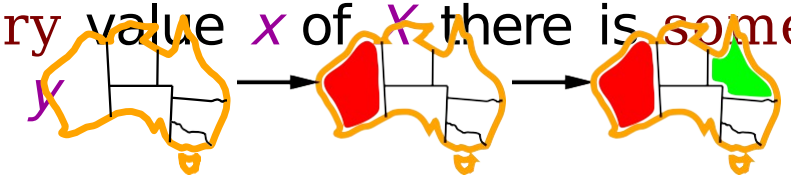


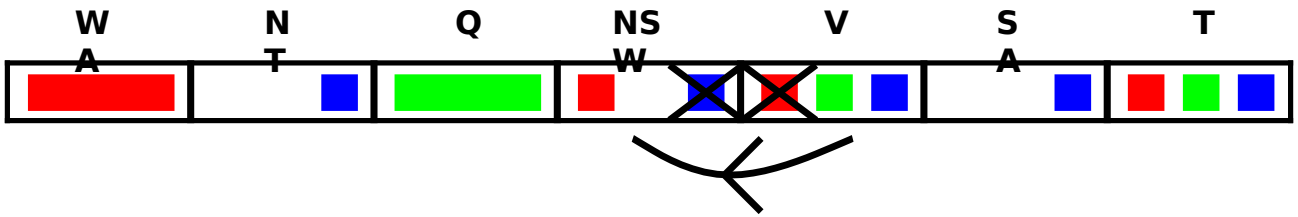
© 2021 Pearson Education Ltd.

Arc consistency

Simplest form of propagation makes each arc consistent

$X \rightarrow Y$ is consistent iff
for every value x of X there is some allowed y





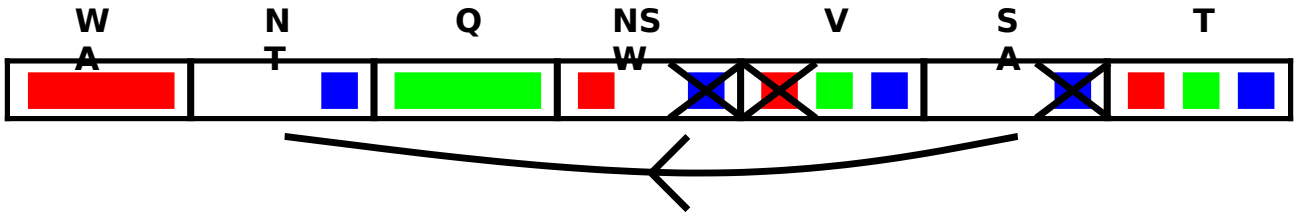
If X loses a value, neighbors of X need to be rechecked

© 2021 Pearson Education Ltd.

Arc consistency

Simplest form of propagation makes each arc consistent

$X \rightarrow Y$ is consistent iff
for every value x of X there is some allowed y



If X loses a value, neighbors of X need to be rechecked

Arc consistency detects failure earlier than forward checking Can be run as a preprocessor after each assignment

© 2021 Pearson Education Ltd.

Arc consistency algorithm

```
function AC-3( csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables {X1, X2, . . . , Xn}
  local variables: queue, a queue of arcs, initially all the arcs in csp
  while queue is not empty do
    (Xi, Xj) ← Remove-First(queue)
    if Remove-Inconsistent-Values(Xi, Xj)
      then for each Xk in Neighbors[Xi] do
        add (Xk, Xi) to queue

function Remove-Inconsistent-Values( Xi, Xj)
  returns true iff succeeds
  removed ← false
  for each x in Domain[Xi] do
    if no value y in Domain[Xj] allows (x,y) to
      satisfy the constraint Xi
```


© 2021 Pearson Education Ltd.

Local Search for CSPs

Local search algorithms can be very effective in solving many CSPs.

Local search algorithms use a complete-state formulation where each state assigns a value to every variable, and the search changes the value of one variable at a time.

Min-conflicts heuristic: value that results in the **minimum number**
of **conflicts** with other variables that **brings us closer to a solution.**

- Usually has a series of **plateaus**

Plateau search: allowing sideways moves to another state with the same score.

- can help local search find its way off the plateau.

Constraint weighting aims to concentrate the search on the important constraints

- Each constraint is given a numeric weight, initially all 1.
- weights adjusted by incrementing when it is violated by the

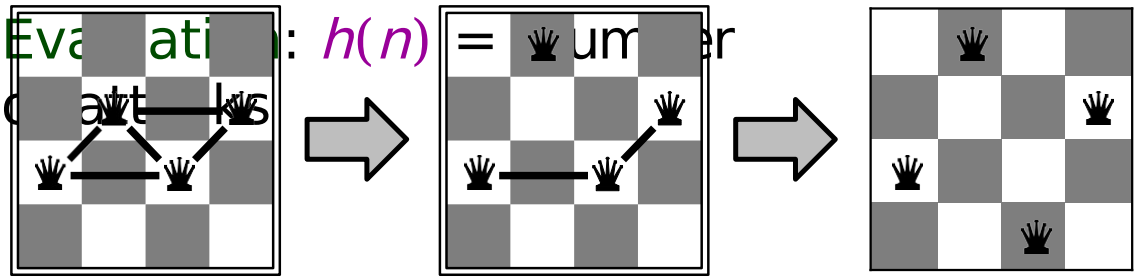
Example: 4-Queens

States: 4 queens in 4 columns ($4^4 = 256$ states)

Operators: move queen in column

Goal test: no attacks

Evaluation: $h(n)$ = number of attacks



h = 5 h = 2 h = 0

© 2021 Pearson Education Ltd.

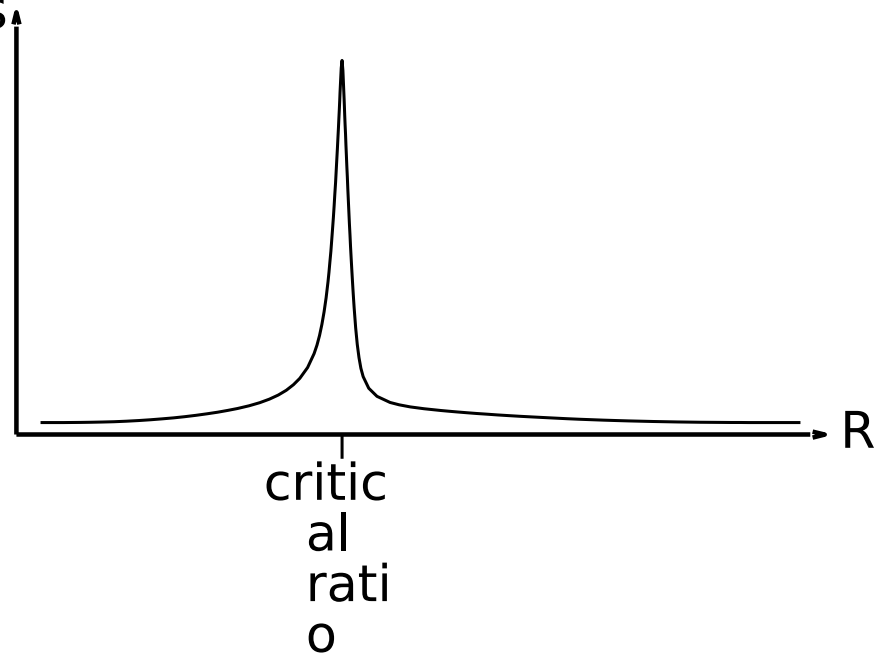
Performance of min-conflicts

Given random initial state, can solve n-queens in almost constant time for arbitrary n with high probability (e.g., n = 10,000,000)

The same appears to be true for any randomly-generated CSP

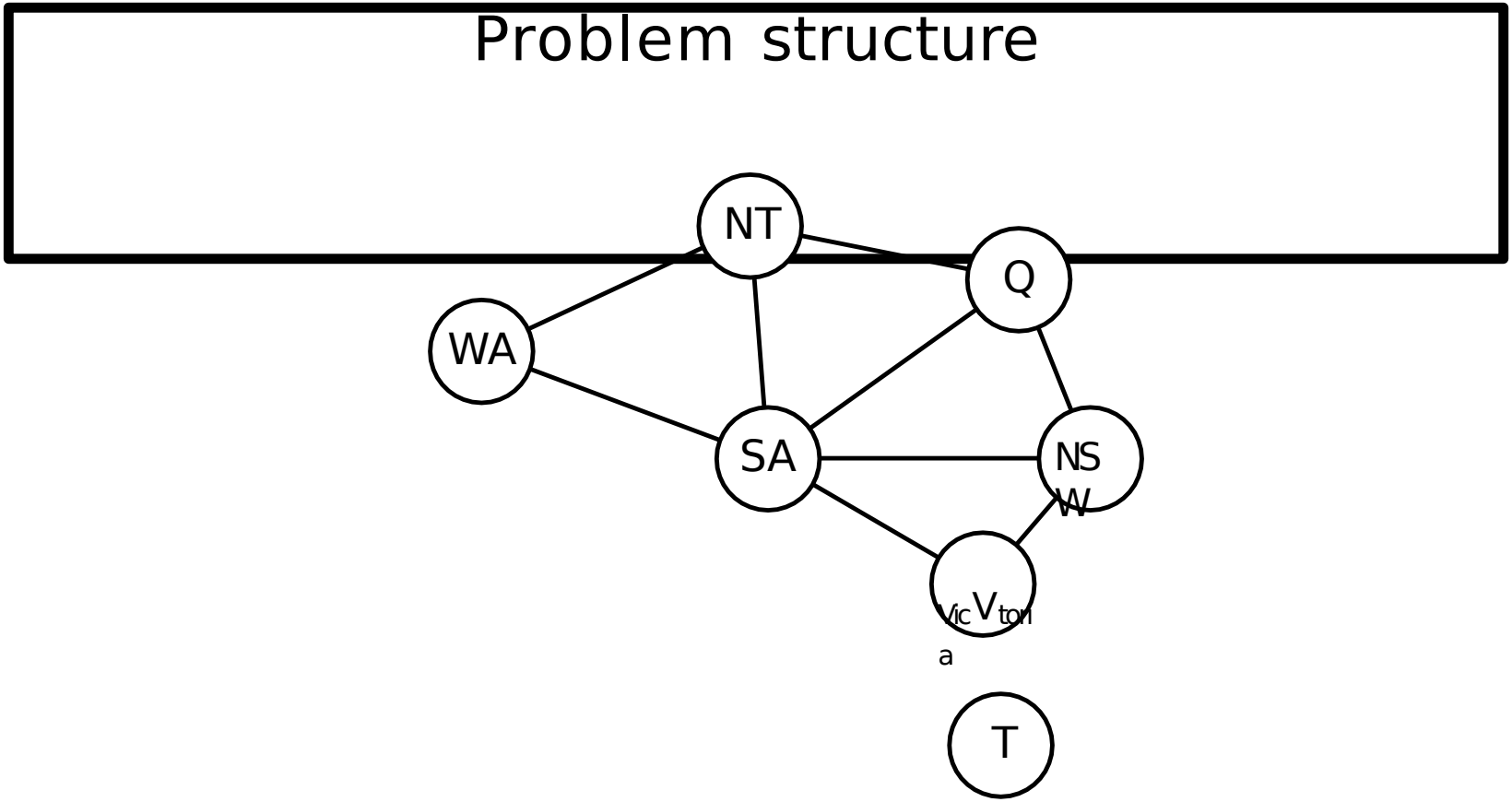
except in a narrow range of the ratio

$$R = \frac{\text{number of constraints}}{\text{number of variables}}$$



© 2021 Pearson Education Ltd.

Problem structure



Tasmania and mainland are independent subproblems Identifiable as connected components of constraint graph

© 2021 Pearson Education Ltd.

Problem structure contd.

Suppose each subproblem has c variables out of n total

Worst-case solution cost is $n/c \cdot d^c$, linear in n

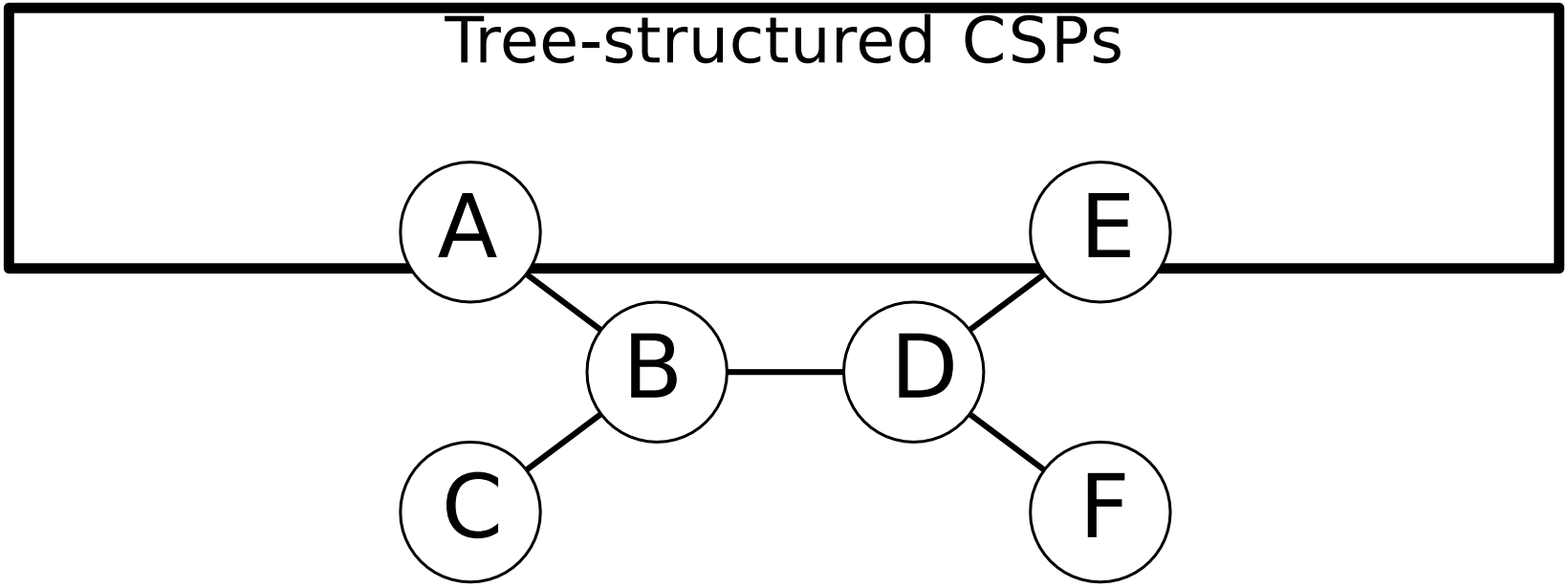
E.g., $n = 80$, $d = 2$, $c = 20$

2^{80} = 4 billion years at 10 million nodes/sec

$4 \cdot 2^{20}$ = 0.4 seconds at 10 million nodes/sec

© 2021 Pearson Education Ltd.

Tree-structured CSPs



Theorem: if the constraint graph has no loops, the CSP can be solved in $O(n\, d^2)$ time

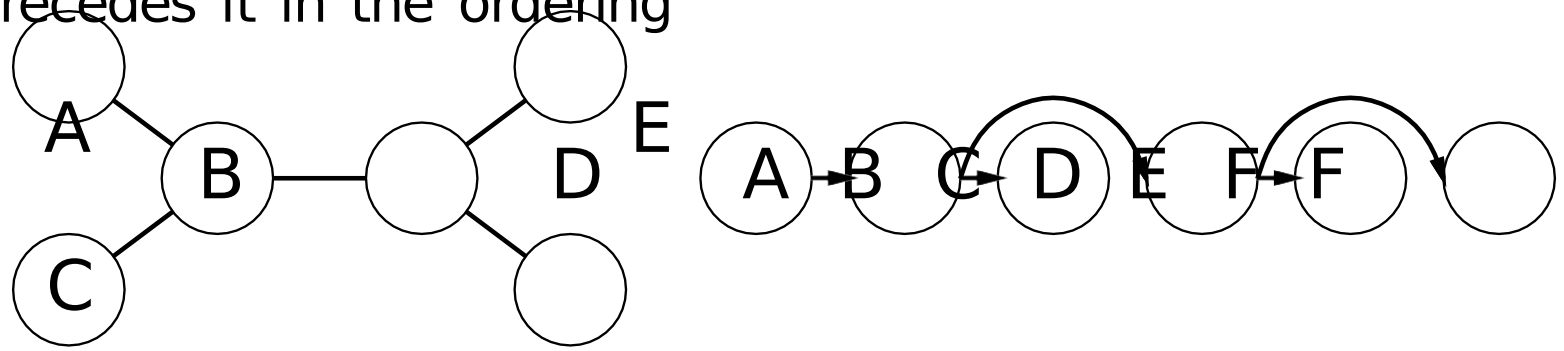
Compare to general CSPs, where worst-case time is $O(d^n)$

This property also applies to logical and probabilistic reasoning:
an important example of the relation between syntactic restrictions and the complexity of reasoning.

© 2021 Pearson Education Ltd.

Algorithm for tree-structured CSPs

1. Choose a variable as root, order variables from root to leaves such that every node's parent precedes it in the ordering



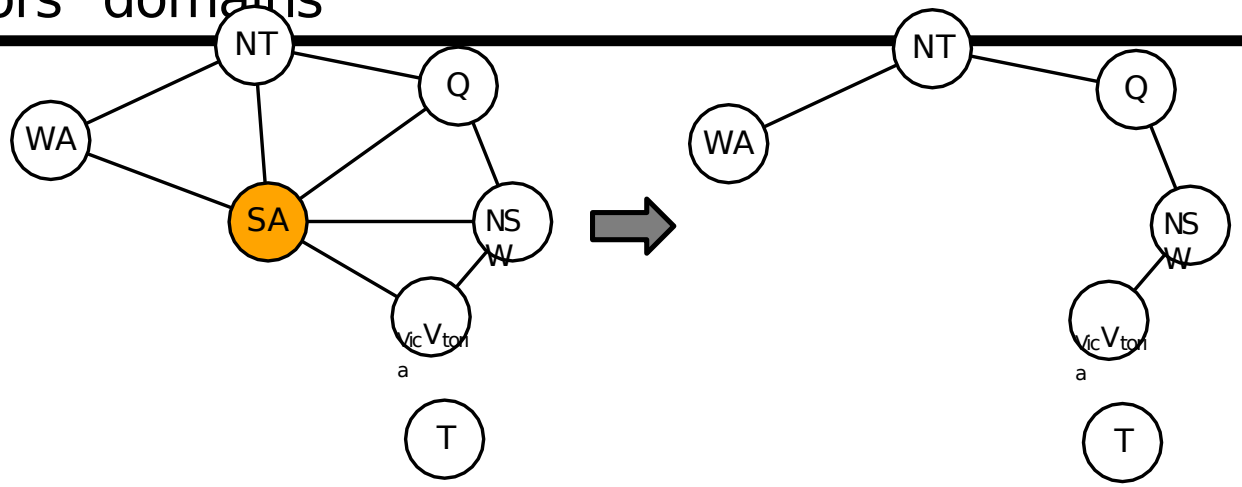
2. For j from n down to 2, apply RemoveInconsistent($Parent(X_j)$, X_j)

3. For j from 1 to n, assign X_j consistently with $Parent(X_j)$

© 2021 Pearson Education Ltd.

Nearly tree-structured CSPs

Conditioning: instantiate a variable, prune its neighbors' domains



Cutset conditioning: instantiate (in all ways) a set of variables such that the remaining constraint graph is a tree

Cutset size c ⇒runtime $O(d^c \cdot (n - c)d^2)$, very fast for small c

© 2021 Pearson Education Ltd.

Iterative algorithms for CSPs

Hill-climbing, simulated annealing typically work with "complete" states, i.e., all variables assigned

To apply to CSPs:
allow states with unsatisfied constraints operators reassign variable values

Variable selection: randomly select any conflicted variable Value selection by min-conflicts heuristic:
choose value that violates the fewest constraints
i.e., hillclimb with $h(n)$ = total number of violated constraints

© 2021 Pearson Education Ltd.

Summary

CSPs are a special kind of problem:
states defined by values of a fixed set of variables goal test defined by constraints on variable values

Backtracking = depth-first search with one variable assigned per node Variable ordering and value selection heuristics help significantly Forward checking prevents assignments that guarantee later failure Constraint propagation (e.g., arc consistency) does additional work to constrain values and detect inconsistencies

Local search using the min-conflicts heuristic has also been applied to constraint satisfaction problems with great success

The CSP representation allows analysis of

© 2021 Pearson Education Ltd.